Technical Report

Rapport Technique

1983/12

TR-SYS-001
NRC NO. 22939

ANEVENT – AN INTERACTIVE COMPUTER MODELLING PACKAGE BASED ON DISCRETE EVENT SIMULATION: PASCAL VERSION PROGRAM DOCUMENTATION

Carl Angers, P.W.U. Graefe

Division of
Mechanical Engineering

Division de
génie mécanique

National Research Council Canada
Conseil national de recherches Canada

Canada

88-04470/

DIVISION OF MECHANICAL ENGINEERING PUBLICATIONS

DM **(Division of Mechanical Engineering Report)**
Scientific and technical information considered important, complete and a lasting contribution to existing knowledge.

TR **(Technical Report)**
Information less broad in scope but a substantial contribution to existing knowledge.

CTR **(Controlled/Classified Technical Report)**
A Technical Report with controlled distribution for national security, proprietary or other reasons.

LM **(Laboratory Memorandum)**
Preliminary or exploratory information with controlled distribution.

CAT **(Calibration Analysis and Test Report)**
Information on minor laboratory projects or services.

PUBLICATIONS DE LA DIVISION DE GÉNIE MÉCANIQUE

DM **(Rapport de la Division de génie mécanique)**
Informations scientifiques et techniques jugées importantes, complètes et susceptibles de contribuer de façon durable à l'avancement des connaissances courantes.

TR **(Rapport technique)**
Informations de moindre importance, mais pouvant contribuer substantiellement à l'avancement des connaissances actuelles.

CTR **(Rapport technique à diffusion contrôlé/classifié)**
Rapport technique à diffusion contrôlée pour des raisons de sécurité nationale, de propriété intellectuelle et autres.

LM **(Mémoire de laboratoire)**
Informations préliminaires ou de nature exploratoire à diffusion contrôlée.

CAT **(Rapport d'étalonnage d'analyse et d'essai)**
Informations sur de petits projets ou des services de laboratoire.

UNLIMITED
UNCLASSIFIED

ILLIMITÉ
NON CLASSIFIÉ

ANEVENT - AN INTERACTIVE COMPUTER MODELLING PACKAGE BASED ON DISCRETE EVENT SIMULATION: PASCAL VERSION

PROGRAM DOCUMENTATION

ANEVENT - UNE SÉRIE DE PROGRAMMES POUR MODÉLISATION INFORMATISÉE ET INTERACTIVE BASÉE SUR LA SIMULATION PAR EVÈNEMENTS DISCRETS: VERSION PASCAL

Carl Angers*, P.W.U. Graefe
*Co-Op Student, University of Waterloo

Technical Report

1983/12

Rapport technique

TR-SYS-001
NRC NO. 22939

P. Hamill, Head/Chef
Systems Laboratory/
laboratoire des systèmes

E.H. Dudgeon
Director/
Directeur

ABSTRACT

This report describes the Pascal version of ANEVENT: an interactive computer-modelling package based on discrete event simulation. It includes descriptions of all the routines in the package, programming considerations when using ANEVENT and notes on transfering the package to another computer. The major differences between the FORTRAN and Pascal versions are also documented.

RÉSUMÉ

Ce rapport décrit la version Pascal d'ANEVENT: une série de programmes pour modélisation informatisée et interactive basée sur la simulation par évènements discrets. Les descriptions des sous-programmes sont inclues ainsi que les considérations de programmation pour utiliser ANEVENT et pour l'exécution d'ANEVENT sur d'autres ordinateurs. Les grandes différences entre les versions FORTRAN et Pascal sont aussi traitées.

CONTENTS

		Page
	ABSTRACT	(ii)
	ILLUSTRATIONS	(iv)
	PROGRAM LISTINGS	(iv)
	TABLES	(iv)
1.0	INTRODUCTION	1
2.0	GLOSSARY OF TERMS	2
3.0	MODEL OPERATION	3
4.0	DESCRIPTION OF MODEL MODULES	6
	4.1 Executive Module	9
	4.2 Initialization Module	11
	4.3 Command Interpretaton Module	13
	4.4 Information and Report Writing Module	15
	4.5 Save and Recall Module	16
	4.6 Submodel Events Module	16
	4.7 Library	17
	4.7.1 GASP file manipulation	18
	4.7.2 ASCII string manipulation	19
	4.7.3 Random number generation	20
	4.7.4 Histogram generation	21
	4.7.5 Determination of simulation time	21
5.0	PROGRAMMING CONVENTIONS AND ASSIGNMENTS	21
	5.1 Structure	21
	5.2 Subroutine Check List	21
	5.3 Input - Output	27
	5.4 File Assignment	29
	5.5 Global Variables	29
	5.6 Compilaton	29
6.0	CREATION OF ANEVENT LIBRARY	38
7.0	DATA FOR GLOBAL VARIABLES	38
8.0	TRANSFER OF ANEVENT PACKAGE TO ANOTHER COMPUTER	40
	8.1 Compilation	40
	8.2 Random Number Generation	40
	8.3 I/O Device initialization	40
	8.4 Reading Time of Day and Date	40
	8.5 Operator Interrupts and Time Keeping	40
	8.6 Non-Standard Functions and Procedures	43

CONTENTS (Cont'd)

		Page
9.0	DIFFERENCES BETWEEN THE FORTRAN AND PASCAL VERSIONS	43
10.0	REFERENCES	44
	DOCUMENTATION PAGE	45

ILLUSTRATIONS

Figure

3.1	Operation of Interactive Discrete Event Models	5
3.2	Operation of Non Interactive Discrete Event Models	7
4.1	High Level Structural View of Model	8
4.2	Program Module Hierarchy	10
4.3	Command Interpretation	12

PROGRAM LISTINGS

List

5.5 (a)	Main Program ANEVENT	30
5.5 (b)	Include File: SYSTEM.ANE	31
5.5 (c)	Include File: USER.ANE	33
5.5 (d)	Include File: ROUTINES.ANE	34
6.1 (a)	Command Procedure to Create ANEVENT Library	39
6.1 (b)	Command Procedure to Compile Routines in ANEVENT Library	39
7.0	Data File: ANECOM.DAT	41

TABLES

Table

5.2	Routines in the ANEVENT Package	22
5.3.1	Sense Switch Assignments	28
5.3.2	Input-Output Device Category Assignment	28
7.0	Global Variable Initialization	42

ANEVENT - AN INTERACTIVE COMPUTER MODELLING PACKAGE BASED ON DISCRETE EVENT SIMULATION: PASCAL VERSION PROGRAM DOCUMENTATION

1.0 INTRODUCTION

For a number of years the Division of Mechanical Engineering of the National Research Council has been collaborating with industry in the modeling and simulation of industrial plants. Companies involved included members from the metal refining, mining and transportation industries. The models developed addressed themselves to investigations of current plant operations, equipment evaluations, impact of proposed plant expansions or the design of new plants or transportation systems.

These plants or systems were modeled as either discrete processes or a mixture of discrete and continuous processes.

For the latter type of models a hybrid computer was used. Most of the models were interactive models, permitting model operators to observe the model state evolve in time and enabling them to issue commands to the models, thus directing the path that these simulations were to take.

Out of these activities and the simulation language GASP II [Ref. 1] the ANEVENT simulation package (ANEVENT = ANalysis laboratory discrete EVENT) has evolved. It provides a structure for building interactive, discrete event computer models and provides a number of problem independent subroutines to be used in the model building.

This document describes the overall structure of these discrete event models, interactive or otherwise, and documents the purpose and use of all routines available with the ANEVENT simulation package.

For information on how to build interactive discrete event models the reader is referred to 'ANEVENT: An Interactive Computer Modelling Package Based on Discrete Event Simulation - A User's Guide' [Ref. 3] which describes the modeling steps necessary and illustrates the entire model building procedure by means of the development of a sample model.

Listings and/or copies of the program on 9 track magnetic tape can be obtained for a moderate charge to cover materials and handling by writing to the Systems Laboratory at the address on the Documentation page (p. 45).

2.0 GLOSSARY OF TERMS

To facilitate discussions in the remainder of this document, the following terms which are used quite frequently are defined.

Model A real process or system is analysed and an appropriate understanding of it is realized in the form of a SYSTEM MODEL.

Once it is accepted that this SYSTEM MODEL describes reality adequately for the purpose for which the model is needed, it is then time to plan how the SYSTEM MODEL can be implemented on a computer. The programming specifications required for this implementation are referred to as the COMPUTER MODEL. In this document, unless otherwise specified, a reference to MODEL refers to the computer model.

Discrete Event Model A discrete event model represents reality by a model whose dependent variables change discretely, and only at specified instances of simulated time.

Event The instances of time at which some or all dependent model variables change are called events.

Event Time The time at which an event occurs is referred to as the event time.

Synchronous Mode In the synchronous mode of model operation simulated time between event times is linearly related to real time through a specified speed-up factor.

Asynchronous Mode In the asynchronous mode of model operation simulated time is instantaneously advanced from one event time to the next.

Entity The real process or system to be modeled is thought to be made up of a number of entities, whose interaction produces the process or system behaviour. Entities may thus be 'devices' such as an elevator, train, furnace etc. or 'materials' such as ore, liquid steel, crates or passengers.

Attribute Each entity can be described by a number of attributes of the entity. Device attributes may for instance describe the state of the device, (idle, busy, under repair, etc), the time at which it entered the current status, the size, location, weight or temperature of the device. Similarly material attributes may describe the grade, specific gravity, temperature and dimension of a unit of the material. Events also have attributes associated with them to facilitate identification of the type of event, event time and other parameters required in executing the event. For example, the event of an elevator

arriving at a given floor going up could have the following five attributes: 1. event time, 2. event identification, 3. elevator number, 4. floor number, 5. up/down indicator.

Files Apart from data files which have been prepared externally and are read by the program in order to initialize the model, two types of files are used in the ANEVENT package, GASP files and Model State files. GASP files permit the user to store a number of variables in an entry of a file and achieve an automatic ordering of these entries in the file according to predefined rules.

Model state files are disk files created by the program in response to a SAVE command to save the entire model state. This allows the model user to later initialize the model to any one of these saved states.

Time File File 1 of the GASP files is referred to as the time file. It has one entry for every event that has been scheduled. The entries are ordered in chronological order of event times.

File Entries Each GASP file consists of a number of file entries. Each entry is made up of a number of attributes.

Ranking of File Entries Ranking defines the ordering of entries in a GASP file. For each file a ranking attribute and a ranking rule is specified. For example, the time file (file 1) which contains as entries a list of events to be executed in chronological order is ranked on attribute 1, the event-time, using the low-value-first (LVF) ranking rule, thus ensuring that the first entry in the file corresponds to the next event to be executed. For each of the other GASP files the user is free to choose any attribute for ranking and either one of the two ranking rules, low-value-first or high-value first (HVF).

GASP General Activity Simulation Program. The name of a Fortran based discrete event simulation language developed at US STEEL. Extensions to this language resulted in GASP II and GASP IIA [Ref. 1] and the combined discrete-continuous simulation language GASP IV [Ref. 4]. ANEVENT is based on GASP II in that it uses the same file manipulator mechanism.

3.0 MODEL OPERATION

The ANEVENT package provides the user with a choice of running the model in the synchronous mode or asynchronous mode through the specification of a model mode variable.

In the synchronous mode the system state is generally displayed through CRT messages or graphics for the benefit of the model operator. This is the mode used to evaluate operating strategies in

great detail in an interactive fashion, with most of the decision making being done by the model operator.

The asynchronous mode is usually selected in cases where all decision making is built into the model and the interest lies in production of performance statistics over long periods of time. A direct display of the model state during simulation is usually not asked for, although periodic snapshots of model state may be gathered on a disk for later examination. The model mode may be changed during the simulation under operator or program control.

The operation of interactive discrete event models using the ANEVENT package is illustrated in Figure 3.1 and consists of the following steps.

1 Model initialization ensures that all model variables and parameters are set to reflect the initial state of the system. It includes the filing of entries into the time file (GASP file 1) which contains particulars of all time events whose event times are known at the start of a simulation run. At least one of these events has to be in the file, and since the events filed are stored in chronological order the first entry in the file corresponds to the event to be executed first.

2 The time at which the next event is to be executed is established by examining the time of the event at the top of the time file.

3 If the simulation mode is synchronous, proceed with step 4, otherwise proceed directly with step 10.

4 A computer clock is set to generate a timer interrupt to occur after an interval of time has elapsed. This interval corresponds to the time difference between the next event time and the current simulation time, suitably scaled by a specified speed-up factor.

5 The program suspends all further action until either a timer interrupt or a model operator interrupt occurs.

6 The model operator can interrupt the simulation process at any time by pressing a special key (the RETURN key in the VAX 11/780 version) in order to issue commands to the model.

7 The type of interrupt that has occurred is examined. If only the operator interrupt was detected, proceed with 8. For a timer interrupt or both interrupts occurring simultaneously proceed with 9.

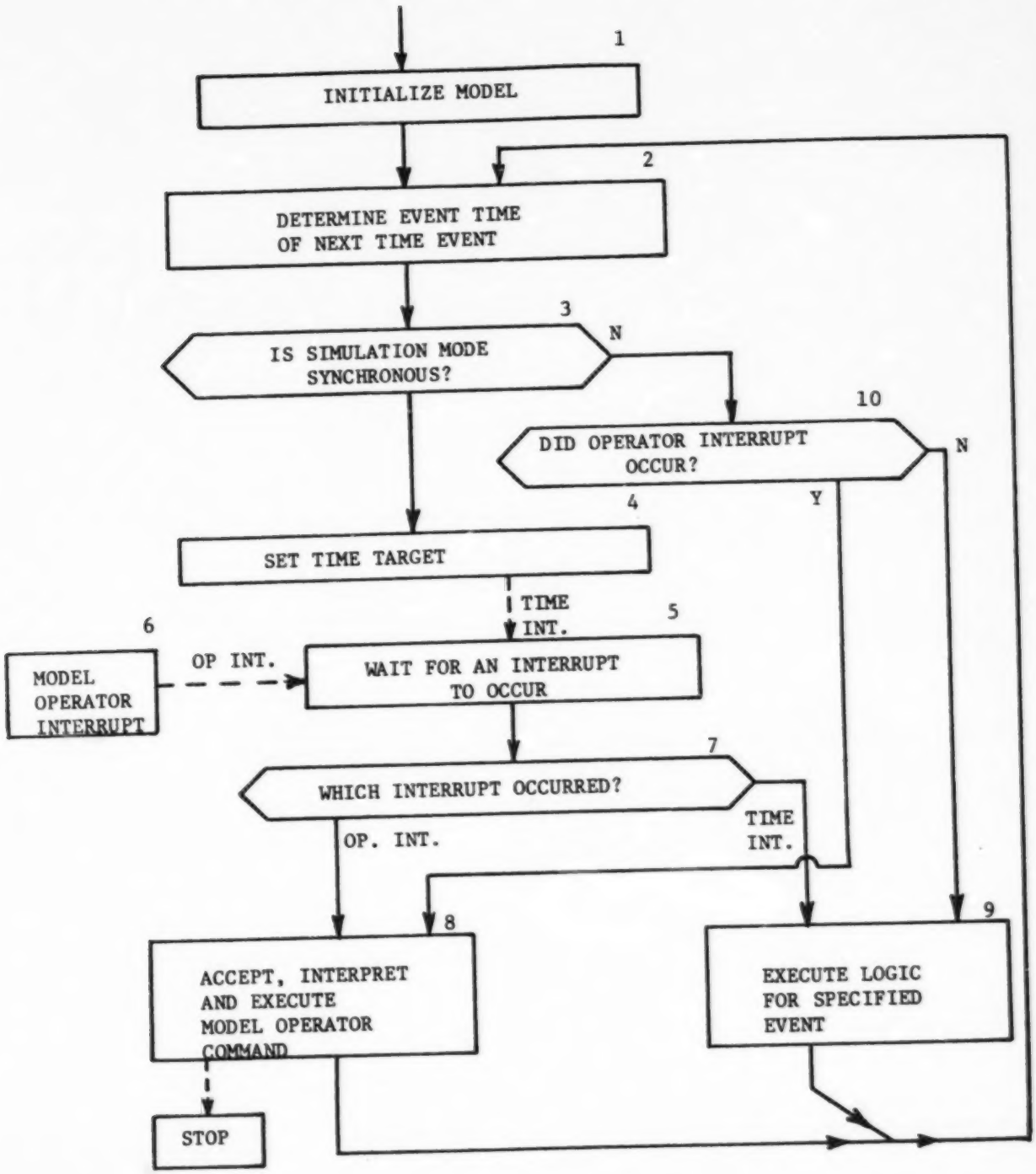


FIG. 3.1 OPERATION OF INTERACTIVE DISCRETE EVENT MODELS

8 The model operator is prompted for input, his commands are accepted, interpreted and executed. Depending on the nature of each command, execution may entail an update of model variables or parameters, a printout of information, the filing of new events into the time file or terminating the simulation run. At the completion of the command input session, proceed with 2 unless the simulation run has been terminated by a QUIT command from the operator.

9 The logic of the specified event is executed. Inputs to the event routine are provided by the event attributes and possibly other variables or parameters of the data base. At completion of the event execution proceed with 2. Note that for the case of simultaneous interrupts from the timer and operator only the timer interrupt has been serviced. The operator interrupt will be serviced only after control has again passed through steps 2 to 7.

10 If an operator interrupt occurred, proceed with step 8, otherwise execute step 9.

It should be noted that the computer clock is not running while an event is being executed or operator commands are being processed. During these times model time remains frozen.

Some models require no operator intervention and are only run in the asynchronous mode. For these models the model structure is simplified considerably in that steps 3 to 8 are not required. Figure 3.2 shows model operations in this case. Here a simulation run is terminated via an end-of-simulation event which is filed in the time file at the start of simulation, its event time being based on run length specifications provided during initialization.

4.0 DESCRIPTION OF MODEL MODULES

To implement the model operations described in the previous section the model is divided into a number of functional modules and data modules. This structure lends itself readily to the use of program overlays on machines with limited memory size for a computational task. A high level view of this structure is shown in Figure 4.1.

An executive module maintains control over the other functional modules for:

1 Initialization
2 Command Interpretation
3 Information and Report Writing
4 Saving and Recalling of Entire Model State
5 Executing Submodel Events.

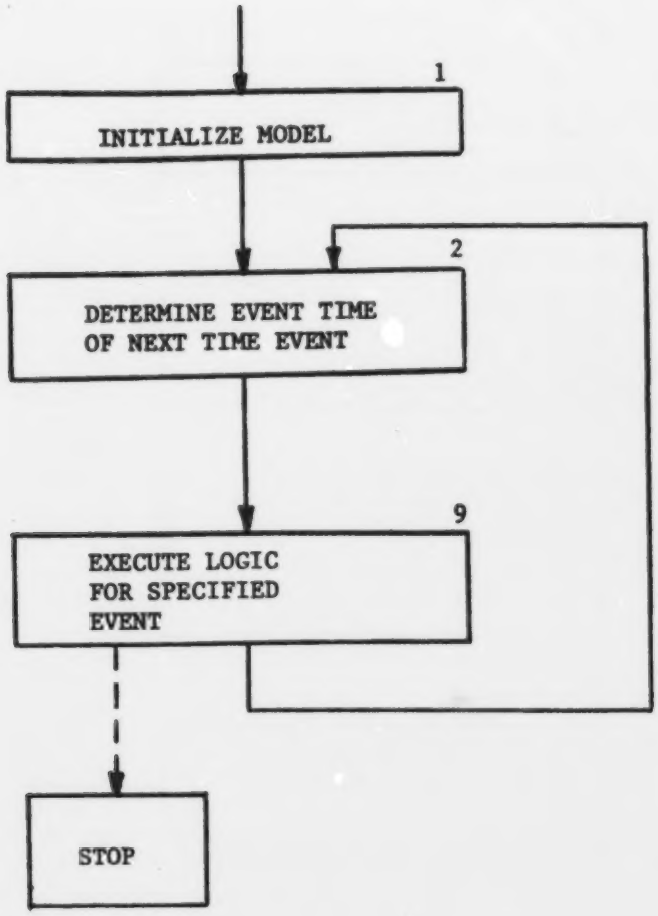


FIG. 3.2 OPERATION OF NON INTERACTIVE
DISCRETE EVENT MODELS

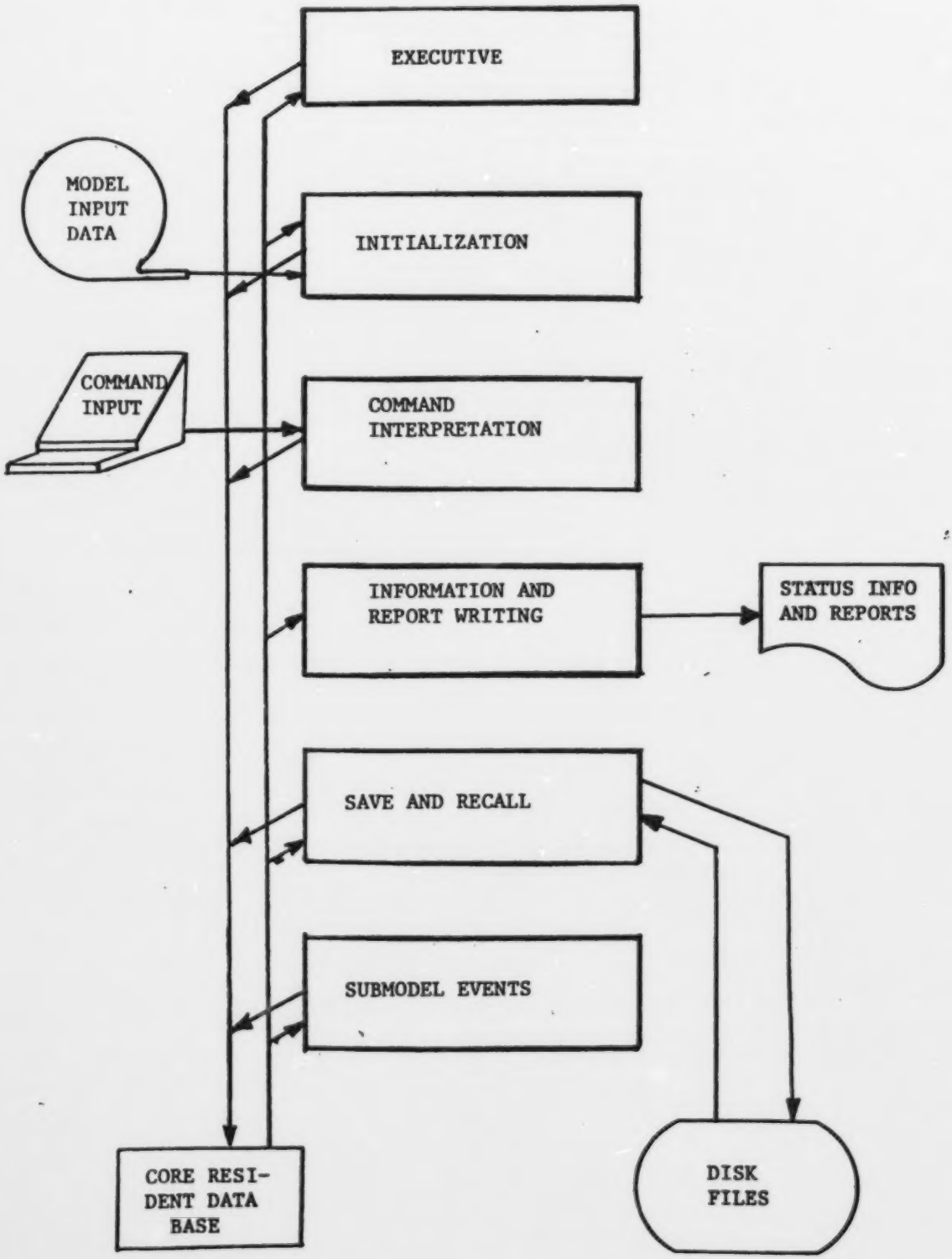


Fig. 4.1 HIGH LEVEL STRUCTURAL VIEW OF MODEL

Data concerning model state, model and system parameters, jobs to be performed etc. is stored in data modules such as:

A Model State Files
B GASP Files
  Time File
  Other User defined files
C Submodel Constants and Variables
D General System Constants and Variables

Initial model status, system constants and variables are stored as disk data files which are read by the initialization module. Operator commands are processed by the command interpretation module. The information and report writing module provides model status information in response to program or operator request. The description of the behaviour of the submodels, structured in terms of a set of discrete events is contained in the functional module 'Submodel Events'.

Each one of the model state files contains a complete set of model constants and variables, system constants and contents of all the GASP files. A model state file is created by the SAVE portion of the Save and Recall Module which copies all current information from the other data modules into the specified model state file. The RECALL portion of the save and recall module performs the complimentary function of updating all data modules according to the content of the specified model state file.

The GASP files consist of a time file which was mentioned in sections 2 and 3, and any number of problem specific inventories, for example station and train files to describe transit passengers in stations and on trains of a transit system etc. These GASP files can be accessed by all functional modules.

Submodel Data can be accessed by all functional modules. The 'General System Constants and Variables' data module receives its initial input from the initialization module and is accessed by the Command Interpretation, the Save and Recall and the Executive modules.

Details of the structure of the various data modules will be given in Section 5.5.

Descending to another level of detail, Figure 4.2 shows a number of program modules within each functional module and their hierarchial relationship in the model. A brief account of each program module will now be given. Section 5 will provide further detail as to subroutine names and calling arguments used.

4.1 Executive Module

The executive module controls the operation of the model as was described in Section 3. It consists of the main program module EX-0 and subprogram modules EX-1 to EX-5.

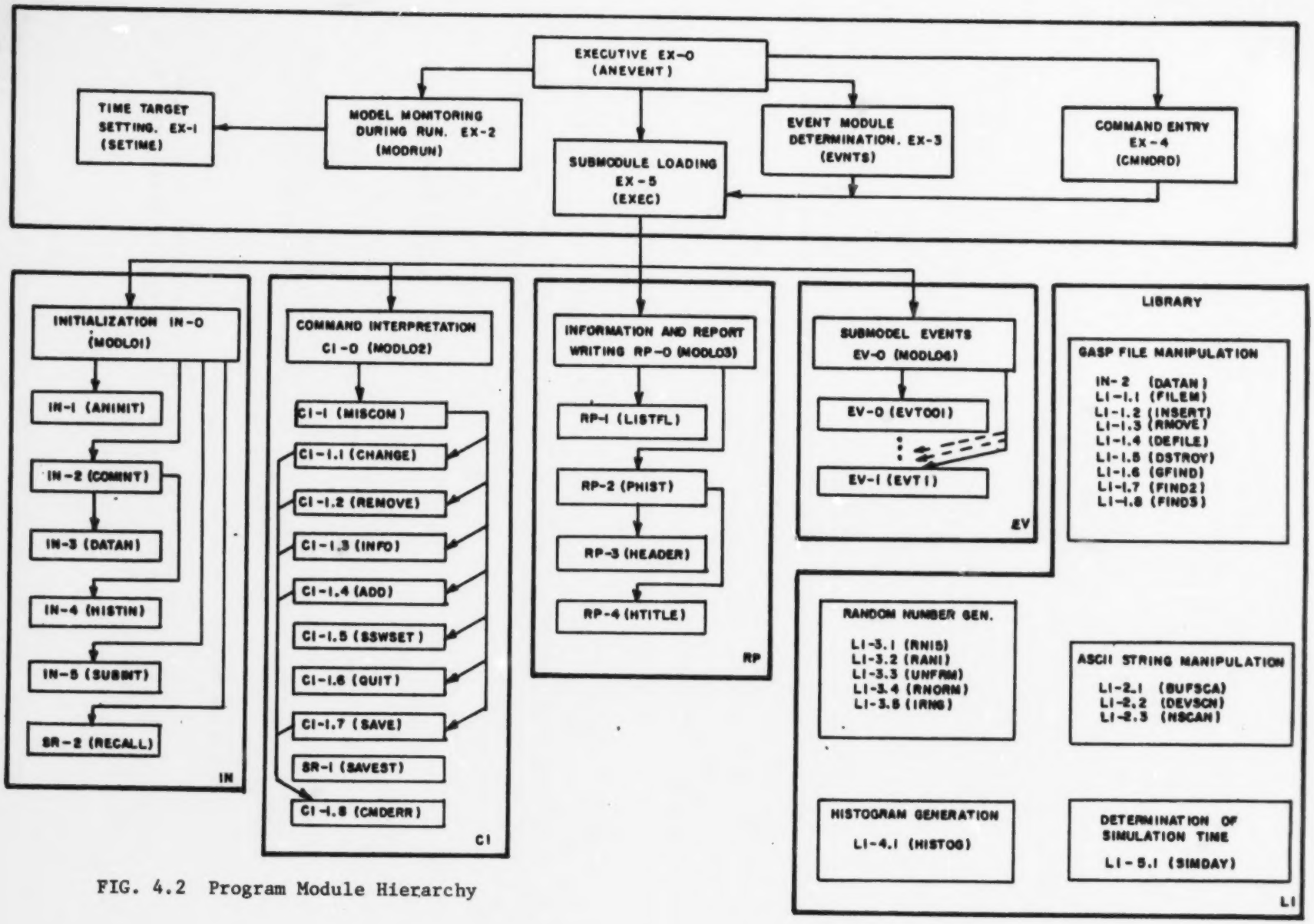


FIG. 4.2 Program Module Hierarchy

If the non-interactive asynchronous form of the ANEVENT package is used, program modules EX-1, EX-2 and EX-4 are not required. EX-0, the main program calls other routines for initialization, target time setting, interrupt monitoring, command processing and event determination and execution.

EX-1, routine SETIME, is the module which sets an interval clock to the time interval between now and the next event time, suitably scaled by a speed-up factor, and starts the clock.

EX-2, routine MODRUN, the model monitoring module calls EX-1 to set and start the clock, and then waits for either a timing interrupt or model operator interrupt. Once an interrupt occurs it identifies the type of interrupt and establishes the current simulation time.

EX-3, routine EVNTS, is the module which calls on the appropriate submodel event module which handles the execution of the given event.

EX-4, routine CMNDRD, prompts the user for an input command category, reads this category and transfers control to the command interpretation module CI-0. An input "?" results in a command category menu being displayed. Current categories are 'USER' and 'MISCELLANEOUS'. The model user may replace the 'USER' category by one or more specific categories such as 'TRAINS' and 'STATIONS' say, for a model of a transit system.

EX-5, routine EXEC, handles the loading of overlay modules. In the VAX 11/780 version this routine could be eliminated since overlays are not used. For computer implementations using overlays this routine is useful specifically if one overlay module is to be loaded upon request of another overlay module, following execution of the first overlay module.

4.2 Initialization Module

The initialization module is responsible for initializing the entire model data base either directly from user created disk data files or from a user specified model state file. The main module IN-0 is called either directly from the main program EX-0 or, if overlays are used, from routine EXEC which is module EX-5.

IN-0, routine MODL01, first calls IN-1 to initialize the I/O devices defined by variab'es DEVICE1 to DEVICE20. It then establishes the source of the initializing data. If it originates from a model state file, the recall module SR-2 is called for initialization. Otherwise module IN-2 is called to initialize the system global variables, the GASP filing mechanism, while the user written module IN-4 is called to initialize all user-defined global variables.

IN-1 routine ANINIT reads the file names from disk to be associated with the file variables: DEVICE1 to DEVICE20.

IN-2, routine COMINT reads data from disk data files and zeroes out portions of the system global variables as required. Part of this is done directly while modules IN-2 and IN-3 are called to deal with more specific tasks.

IN-3, subroutine DATAN initialises all parameters to do with the GASP filing mechanism and files particulars for events that are initialized at the start.

IN-4, subroutine HISTIN initialises all histogram parameters.

IN-5, subroutine SUBINT is a user written routine which is responsible for the initialization of all the user defined global variables.

4.3 Command Interpretation Module

The command interpretation module is responsible for prompting the operator for inputs, accepting his input, interpreting the command and then executing it. This is done at two levels as illustrated in Figure 4.3. At the first level a command category is chosen from one or more user command categories and the miscellaneous command category. Once a category has been chosen the operator will be prompted for commands at the second level until a carriage return signals the end of the command input session.

The main command interpretation module is CI-0 which is called either directly from EX-4, or, if overlays are used, from EX-5.

CI-0, routine MODL02 examines the first character of the specified command category and branches to an appropriate section in this routine to proceed further. The user will have to add appropriate coding for processing all user commands. For the miscellaneous category of commands program module CI-1 is called for further processing.

CI-1, routine MISCOM processes all miscellaneous commands. A menu of these commands is displayed as an initial prompt for each command. The operator response is accepted into a command buffer, the type of miscellaneous command is established and the appropriate module CI-1.1 to CI-1.7 which deals specifically with this command is called.

CI-1.1, routine CHANGE, processes the change command. The change command has the format

CHANGE,NFILE,NENTRY

It displays entry NENTRY of GASP file NFILE and requests the operator to specify the attribute which he wishes to change and its new value. The

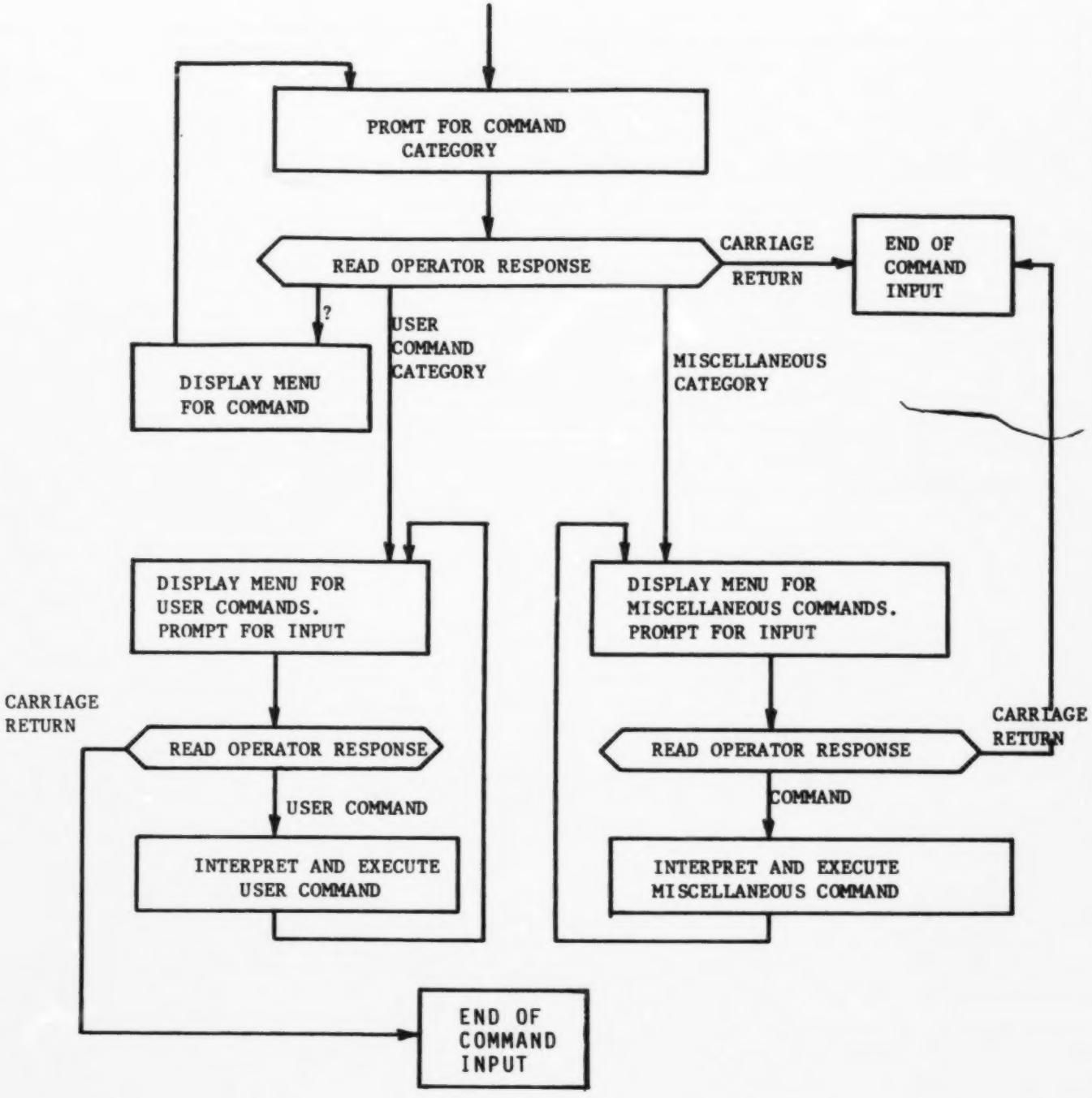


FIG. 4.3 COMMAND INTERPRETATION

operator is prompted for pairs of attribute numbers and values until a zero is entered for an attribute number. In that case a final version of the entry is displayed and the CHANGE command process is terminated.

CI-1.2, routine REMOVE processes the remove command. It has the format

REMOVE,NFILE,NENTRY

and removes entry NENTRY from GASP file NFILE.

CI-1.3, routine INFO provides information on a specified device. The command

INFO,DMNEM,DEVNUM

specifies the device by device mnemonic DMNEM and device number DEVNUM. The device mnenomic is coded into a device code by routine DEVSCN from the library module, and then device code and number are passed to the Information and Report Writing module RP-0 for a printout of the desired information. Any user defined device mnemonics will have to be declared in DEVSCN and coding for information type out will be required in module RP-0.

CI-1.4, routine ADD allows the model operator to add an entry to a specified GASP file NFILE by issuing the command

ADD,NFILE

In response to this command the model operator is asked for up to IMZZ attributes for the entry to be filed. IMZZ is a parameter that is defined in the main program ANEVENT.

CI-1.5, routine SSWSET gives the model operator the opportunity to set or reset any or all of the 50 sense switches by issuing the command

SENSE

The operator will then be provided with a display showing the current settings of all sense switches and a prompt for a switch number and status. A switch number of zero will terminate the SENSE command. A negative switch number will set all sense switches to the specified status.

CI-1.6, routine QUIT terminates the simulation run in response to the command

QUIT

Currently this routine deassigns the computer flags for the timer and operator interrupts and halts execution. If the user desires to perform

additional functions at this point, such as providing for the printout of summary reports or the closing of output files he has to add appropriate coding to this routine.

CI-1.7, routine SAVE processes the command

SAVE,NSTATE

to save the entire system state including all GASP files into system state file NSTATF. Provided this file was given a number between 1 and 5 program module SR-1 in the functional module Save and Recall is called upon to perform the saving operation.

CI-1.8, routine CMDERR is called by program modules CI-1.1, CI-1.2, CI-1.3, CI-1.4 and CI-1.7 if an error was made by the model operator in entering his command. The message ERROR followed by the misunderstood command is given before returning to the point where the menu for the miscellaneous commands is displayed.

4.4 Information and Report Writing Module

Data concerning the state of any process or device in the model, as well as model constants and statistical data gathered so far during the simulation is available in the model data base. In order to gain access to this data in a format useful to the model user the INFO command is used in the interactive version of the ANEVENT package. The INFO command makes use of the various program modules of this Information and Report Writing module. However, some program modules can also be called directly by user written programs if such a requirement arises. The following program modules are available in this functional module:

RP-0, routine MODLO3 uses the specified device code and number as a key to what information is to be printed out. (The correspondence between device mnenomic and device code is defined in routine DEVSCN). The routine currently provides information on the following:

DEVICE CODE	DEVICE
1	Entire model
2	GASP files
13	Histograms

Up to 20 device codes are allowed for at present. The user may define additional devices and provide coding for the desired output. For device code 1 information on all other devices is provided. For device code 2 the content of the specified GASP file is displayed with its entries in a logical order by calling program module RP-1. Device code 13 requests a display of the specified histogram via program module RP-2. If no device number was specified information on all members with the specified device code is displayed.

RP-1, routine LISTFL displays the content of the specified GASP file in the order of the ranking rule specified for that file.

RP-2, routine PHIST displays the specified histogram. It in turn makes use of program modules RP-3 and RP-4 to obtain an appropriate simulation run header and histogram title.

RP-3, routine HEADER may be called whenever a simulation run identification is needed on any simulation output report. It advances the output device to the top of a new page and then prints the date and time at which the current simulation run was started. The model user may add coding to write an appropriate text to identify this particular simulation run.

RP-4, routine HTITLE is used to provide a histogram number and title printout for the histogram printed by routine PHIST, program module RP-2. The user has to add coding to this routine to specify his histogram titles.

4.5 Save and Recall Module

This module makes it possible for the model operator to save the model state in one of the model state files, or to initialize the model to one of the previously saved states. The module is broken into two program modules, the save module SR-1 and recall module SR-2.

SR-1, In response to the SAVE command routine SAVEST requests up to 80 characters of text from the model operator to describe the model state scenario to be saved. This scenario and the entire model data base is then saved in the specified model state file.

SR-2, routine RECALL is called upon whenever the model operator specifies a RESTART FILE NUMBER during the initialization. It reads all data from the specified model state file into the model data base and provides the model operator with a display of the model state description and the simulation time to which the model has just been initialized. If CRT or graphic displays are used, coding for the regeneration of these displays from the database are to be added.

4.6 Submodel Events Module

This functional module is responsible for the execution of the event whose event time has been reached. Since the coding for the events is problem specific the model user will have to provide most of the coding for this module. The module is structured such that the main module EV-0 checks the validity of the event number EVENT and then calls the appropriate program module EV-i for event i.

EV-0, subroutine MODL06 currently allows for 40 events from event #1 to event #40. For any event number outside this range an error message is displayed.

The user has to insert appropriate call statements to event subprograms written by him. These subprograms are named EVTxyz where xyz are the three digits for event i. If more than 40 events are required, the user has to modify subroutine MODL06 accordingly.

If no overlay is required then routine MODL06 can directly replace routine EVNTS by merely changing its name to EVNTS and deleting the former EVNTS routine.

If in an overlay situation the amount of coding for events exceeds the space for this functional module then further overlay modules MODL07, MODL08 etc. may be created and the events divided up among these overlay modules. Subroutine EVNTS in this case is used to determine the proper overlay module based on the event number.

EV-1, subroutine EVT001 is provided for model implementations on a 16 bit computer. On such a computer it may occur that the value for simulation time which is an integer quantity, or any other model variable representing some absolute simulation time exceeds the maximum value of 32767 that an integer variable can take.

The appropriate version of subroutine FILEM* will in this case execute an event 1 whenvever simulation time approaches this maximum value to within TMAX time units. TMAX is the maximum interval to which the ANEVENT package allows the computer clock to be set.

Event #1 will then subtract 14400 time units from all variables representing simulation time in time units. The user will have to provide coding to perform this on all user defined time variables. The number of such resets is counted by an ANEVENT system variable NRESET.

4.7 Library

The model library contains a number of routines which are used in many of the modules discussed above. In particular they should prove valuable to the user for file manipulation, command processing, random number generation etc.

The library routines fall into several broad groups such as

1 GASP file manipulation
2 ASCII string manipulation
3 Random number generation
4 Histogram generation
5 Determination of simulation time

These will now be described.

4.7.1 GASP file manipulation

The GASP files used in this package are organized as linked lists [see Ref. 1]. Each file forms a linked list. Each file entry is a Pascal record containing an integer array of IMZZ elements where attributes are stored and a pointer variable pointing to the next entry. The entries in a linked list are in order and so the first entry is at the beginning of the list etc.
Array KRANK stores the ranking attributes for each file and array INN specifies the ranking rule. An array NQZZ stores the number of entries in each file. Entries are entered into the file by loading entry attributes into a transfer vector COLUMN and then calling the filing routine FILEM(NFILE) to file the entry into file NFILE. A call to RMOVE removes an entry and copies entry attributes into array COLUMN. The following file manipulation routines are available.
IN-2, routine DATAN initializes the filing system

LI-1.1, subroutine FILEM(NFILE) will file the entry stored in array COLUMN into the specified file NFILE using the ranking attribute and ranking rule specified for this file.

LI-1.2, subroutine INSERT(NFILE,KCOL,IFL) is used to file an entry if it is important where the entry is to be placed in case of a tie in the values of the ranking attributes. Depending on the value of IFL(=1 or =2) the routine will file the entry stored in array COLUMN into the specified file NFILE such that it is logically located before or after the entry at position KCOL.

LI-1.3, subroutine RMOVE (KCOL,NFILE) will remove the entry in position KCOL from file NFILE and copy it into array COLUMN. To remove the first entry in a file, use KCOL=1 and to remove the last entry, use KCOL=NQZZ[nfile] KCOL can also be the value returned by one of the FIND routines.

LI-1.4, subroutine DEFILE (NFILE,NVAL,JATT) will remove and destroy all entries from file NFILE which have the value NVAL in the attribute given by JATT.

LI-1.5, subroutine DSTROY(NFILE) will list all entries in file NFILE and then it will remove and destroy all of these entries.

LI-1.6, subroutine GFIND(NVAL,NFILE,JATT,KCOL,IBF) tries to find the position number KCOL of the first entry of file NFILE which has the value NVAL in the attribute given by JATT. A copy of the entry is returned in array IBF. If no such entry can be found KCOL is returned as zero.

LI-1.7, subroutine FIND2(NVAL1,NVAL2,NFILE,JATT1,JATT2,KCOL,IBF) is similar to GFIND except that the entry to be found has to match values NVAL1 and NVAL2 in the attributes given by JATT1 and JATT2.

LI-1.8, subroutine FIND3(NVAL1,NVAL2,NVAL3,NFILE,JATT1,JATT2,JATT3,KCOL,IBF) is similar to GFIND except that the entry to be found has to match values NVAL1, NVAL2 and NVAL3 in the attributes given by JATT1, JATT2 and JATT3.

4.7.2 ASCII string manipulation

Operator commands are best given in the form of a character string with command fields separated by commas. While some computers provide standard routines to manipulate such character strings, many do not. For this reason a number of ASCII string manipulation routines are included in the library to be used for the extraction of command fields from a command string and the translation of a command field in ASCII form into a valid command code, material code, device code and number or a number. Basically this string manipulation works as follows:

The command to be decoded is stored in ASCII in array INPBUF. A pointer ISCNST points to the element of INPBUF at which the scan for a command field (any string of characters separated by commas or blanks) is to start. The command fields are extracted one at a time into array IWORK. IWORK serves as the input to routines which will compare the command field to prestored fields for actions, materials, devices or numbers in order to assign an appropriate code corresponding to the command field. The following routines are available.

LI-2.1, routine BUFSCA(NBUF) scans array INPBUF between position ISCNST and the next comma or blank. It determines the length NBUF of this command field, copies the field into array IWORK and updates the pointer ISCNST.

LI-2.2, subroutine DEVSCN (N) tries to match the first N characters in array IWORK with a list of prestored device mnemonics. If a match is found an appropriate device code DEVICECODE is generated. If no match is found, the model operator is informed of this and prompted for a correct device mnemonic.

Currently the only devices defined are

Device	Device Mnemonic	Device Code
Entire model	AL	1
GASP files	GA	2
Histograms	HI	13

For other, user defined devices appropriate statements have to be added. If the user requires command field scanning for actions or materials he has to provide similar routines which can easily be derived through copying and slight modifications of routine DEVSCN.

LI-2.3, subroutine NSCAN (IBUF,N,NUMBER) attempts to translate the N digit number contained in ASCII in array IBUF into an integer number

NUMBER. If IBUF does not contain digits NUMBER is set to zero and the global error flag ERROR is set to TRUE.

4.7.3 Random number generation

The library contains a number of routines to generate random numbers from uniform distributions, normal distributions and distributions based on histograms of actual observations. All of these routines are based on the machine specific random number generator RN15 which is written in assembly code for the VAX 11/780. If a different computer is to be used a routine RN15 will have to be written. The following routines are available.

LI-3.1, routine RN15 (IRN) uses the odd, positive random number seed IRN which is less or equal to 32767 and generates a new random number IRN in the range between 1 and 32767.

LI-3.2, function RAN1(ISTRM) uses random number stream ISTRM for seed generation and updating to produce a real random number RAN1 in the range between 0 and 1.

LI-3.3, function UNFRM(A,B,ISTRM) uses random number stream ISTRM for seed generating and updating to produce a real, uniformly distributed random number UNFRM in the range from A to B.

LI-3.4, function RNORM(XMEAN,STDV,ISTRM) uses random number stream ISTRM for seed generating and updating to produce a real, normally distributed random number RNORM with a mean value of XMEAN and a standard deviation of STDV.

LI-3.5, subroutine IRNG (IX,KUMX,NCX,ISTRM,IOUT) is used to generate a random number IOUT from a distribution based on field observations. Suppose that observations on a task time were made as follows:

Task time in time units	Number of Observations
15	29
13	83
18	11
25	5
TOTAL	128

This data is used as input to a program PPROC1 described in Ref 2 to generate a suitably scaled cumulative distribution KUMX of the NCX=4 observed task times of 29,83,11 and 5 stored in array IX. IX,KUMX and NCX are read in during initialization. Subroutine IRNG will use these and the random number stream ISTRM to generate the random number IOUT. In the above example IOUT will have the value 15 29 times out of 128, the value 13 83 times out of 128 etc.

4.7.4 Histogram generation

Histograms are initialized by subroutine HISTIN, updated by subroutine HISTOG described below and printed out by routine PHIST which in turn calls on HTITLE for histogram titles.

LI-4.1, subroutine HISTOG (X,N) is used to build up the histogram by adding another observed real value X to histogram N.

4.7.5 Determination of simulation time

Simulation time is being tracked through the variables SIMTIM and NRESET such that

simulation time = 14400 * NRESET + SIMTIM time units.

To convert to a model time of days, hours, minutes and seconds with a specified starting time for the model the offsets HROFST and IMOFST for hours and minutes have to be incorporated into this conversion. Program module LI-5.1, subroutine SIMDAY(ITIME,IDAY,IHOUR,IMIN,ISEC) performs this task by tansforming the simulation time 14400*NRESET+ITIME into the model time of IDAY days, IHOUR hours, IMIN minutes and ISEC seconds.

Note that on a 32 bit machine simulation time does not have to be reset, and NRESET always has the value of zero.

5.0 PROGRAMMING CONVENTIONS AND ASSIGNMENTS

In this section program organization, conventions and assignment will be discussed.

5.1 Structure

While the ANEVENT package described here runs on a virtual memory machine it can easily be implemented on computers with limited task size by using program overlays. Thus the executive module is assumed to be the root of an overlay tree structure with functional modules IN,CI,RP,EV [Fig. 4.2] constituting one or more overlay modules each. To handle the case of one functional module such as CI calling another one, such as RP as in the case of processing a command 'INFO, GASP', this linking between overlays has to be done through the routine EXEC in the root segment. To convert the current structure to an overlay structure replace the call to MODL01 by a call to EXEC(1) in routines ANEVENT and replace calls to MODL02, MODL03 and MODL06 by calls to EXEC(2), EXEC(3) and EXEC(6) in routines CMNDRD, INFO, and EVNTS respectively.

5.2 Subroutine Check List

Table 5.2 provides an alphabetical list of all routines in the ANEVENT package. It indicates which routines are complete and which ones require some user-generated code.

ROUTINE NAME	UTILITY PROGRAM	USER ADD.REQ.
ADD		
ANINIT		X
BUFSCA	X	
CHANGE		
CMDERR		
CMNDRD		
COMINT		
DATAN		
DEFILE	X	
DEVSCN	X	X
DSTROY	X	
EVNTS		X
EVT001		X
EXEC	X	X
FILEM	X	
GFIND	X	
FIND2	X	
FIND3	X	
HEADER	X	X
HISTIN		
HISTOG	X	
HTITLE		X

ROUTINE NAME	UTILITY PROGRAM	USER ADD.REQ.
INFO		
INSERT	X	
IRNG	X	
LISTFL	X	
MISCOM		
MODL01		
MODL02		X
MODL03		X
MODL06		X
MODRUN		
NSCAN	X	
PHISI		
QUIT		X
RAN1	X	
RECALL		X
REMOVE		
RMOVE	X	
RNORM	X	
RN15	X	
SAVE		
SAVEST		X
SET		
SETIME		
SIMDAY	X	
SSWSET	X	
SUBINT		
UNFRM	X	

TABLE 5.2 Routines in the ANEVENT Package

The following list summarizes the purpose of each routine as well as arguments required and, where applicable, the required function of the user code to be added to the routine

ADD — Responds to the add command 'ADD,NFILE' to add an entry to file NFILE

ANEVENT — Main program for the interactive model, controls simulation run in interactive or asynchronous mode as specified by the user.

ANINIT — Initializes event flags and I/O parameters, called only by main program ANEVENT.

BUFSCA(NBUF)
scans array INPBUF between ISCNST and the next comma or blank and copies the NBUF intermediate characters into array IWORK

CHANGE — Responds to the 'CHANGE,NFILE,NENTRY' command to allow changes to be made to entry NENTRY of GASP file NFILE.

CMDERR — Provides printouts of errors during command decoding

CMNDRD — Accepts command categories from the model operator

COMINT — Initializes global system variables except for I/O parameters which were initialized in ANINIT.

DATAN — Initializes all GASP filing parameters and some GASP file entries

DEFILE(NFILE,NVAL,JATT)
removes all entries from file NFILE with the value NVAL in the JATT'th attribute.

DEVSCN(N)
scans the first N characters of array IWORK and translates them into a device code DEVICECODE. The user may want to define device codes for problem specific devices.

DSTROY(NFILE)
lists and then deletes all entries of GASP file NFILE

EVNTS — passes control to the coding for an event whose number is EVENT. In an overlay environment the user may wish to add coding to determine in which overlay module the coding for the specified event is located.

EVT001 Reduces the simulation time stored in SIMTIM by 14400 time units and increments the nubmer of such resets NRESET by one. 14400 time units are also subtracted from all variables representing a clock time. The user has to add coding to subtract 14400 time units from all such user defined variables

EXEC(N) calls on program module N by calling MODLXX where XX=N directly or arranging for overlay module N to be loaded. All overlay modules will return LINK set to zero unless another overlay module M is to be executed immediately, in which case LINK will be set to M before returning from the overlay module.

FILEM(NFILE)
files the content of array COLUMN as an entry of GASP file NFILE

GFIND(NVAL,NFILE,JATT,KCOL,IBF)
attempts to find the first entry in GASP file NFILE whose JATT'th attribute has the value NVAL. If such an entry can be found, KCOL is set to the column number of the filing array where the entry is stored, and a copy of the entry is made in array IBF. If no such entry can be found KCOL is returned as zero.

FIND2(NVAL1,NVAL2,NFILE,JATT1,JATT2,KCOL,IBF)
is similar to routine GFIND except that the search includes two values NVAL1 and NVAL2 as the JATT1'th and JATT2'th attribute.

FIND3(NVAL1,NVAL2,NVAL3,NFILE,JATT1,JATT2,JATT3,KCOL,IBF)
is similar to routine GFIND except that here 3 values and 3 attributes are involved

HEADER(ID) advances the output device whose code is given by ID to a new page, and prints a user specified header as well as the time and date at which the simulation run was started.

HISTIN Initializes histogram parameters

HISTOG(X1,N)
enters an entry into histogram N which represents the real value of X1.

HTITLE(JH)prints out the title for histogram JH. The user provides appropriate histogram titles through the CASE statement.

INFO Responds to the 'INFO,DMNEM,DEVNUM' command. It interprets the operator command into a device code and number which are passed on to subroutine MODL03 via the global variables DEVICECODE and DEVICENUMBER.

INSERT(NFILE,KCOL,IFL)
inserts into file NFILE the entry stored in array ATRIB.For IFL=1 the entry is stored logically before the entry in column KCOL. For IFL=2 the entry is stored logically after the entry at position KCOL.

IRNG(IX,KUMX,NCX,ISTRM,IOUT)
returns a random deviate IOUT of stream ISTRM generated from the NCX points of the cumulative distribution with co-ordinates stored in arrays IX and KUMX. IOUT takes on values of elements of IX. See reference 2 for preparation of data for IX,KUMX,NCX.

LISTFL(NFILE) lists in logical order the entries of GASP file NFILE.

MISCOM Prompts for, reads, accepts and decodes miscellaneous commands. These commands are CHANGE, REMOVE, INFO, ADD, SENSE, QUIT and SAVE.

MODL01 Initializes the model from either data files or model state files

MODL02 Processes model operator commands by calling MISCOM for miscellaneous commands or executing user provided code for user-defined commands.

MODL03 Provides a printout of information on a modelled device whose device code and number are given by variables DEVICECODE and DEVICENUMBER. For any user-made device definition in routine DEVSCN relevant coding for the information printout has to be provided and inserted in the CASE statement using the DEVICECODE value as the case label.

MODL06 Arranges for a branch to the subroutine which processes the event given by EVENT. For all user defined events XXX a call to subroutine EVTXXX (ie. EVT005 for event 5) has to be inserted in the CASE statement at case label XXX.

MODRUN Starts the model clock and waits for an interrupt. When an interrupt occurs IFLAG is set to 1 for an operator interrupt and set to 2 for a timer interrupt. The interrupt is cleared and simulation time SIMTIM is computed.

In the asynchronous mode of operation (MDLMOD=asynchronous) the setting of the clock and waiting for an interrupt is bypassed. Only the operator interrupt is examined, the timer interrupt is assumed to have occured at all times.

NSCAN(IBUF,N,NUMBER)
translates the N digits of the number contained in array IBUF into the integer number NUMBER. If IBUF does not contain a number, NUMBER is set to zero and an error is indicated by setting the global variable ERROR to TRUE.

PHIST(K) will display histogram K.

QUIT Terminates the simulation run by releasing the flag cluster for the timer and operator interrupts and then stopping. The user may insert further coding in this routine to process problem specific tasks at the termination of the run.

RAN1(ISTRM)
This function routine generates a uniformly distributed random number RAN1 between 0. and 1. from random number stream ISTRM.

RECALL Initializes the model to the state stored in the model state file whose number is given by IFLAG. If CRT or graphic displays are used, coding for the regeneration of these displays from the database are also to be added.

REMOVE responds to the command 'REMOVE, NFILE, NENTRY' by removing the NENTRY'th entry of file NFILE.

RMOVE(KCOL,NFILE)
removes from GASP file NFILE the entry stored in column KCOL of the filing array and copies it into the array ATRIB.

RNORM(XMEAN,STDV,ISTRM)
this function routine generates a normally distributed number RNORM with mean value XMEAN and standard deviation STDV from random number stream ISTRM.

RN15(IRN) with IRN a positive odd integer between 1 and 32767 as a random number seed this routine will generate another uniformly distributed positive odd random number IRN.

SAVE responds to the command 'SAVE,NFILE' by decoding the model state file number, storing it in IFLAG and calling routine SAVEST.

SAVEST saves the model state in the model state file whose number is given by IFLAG.

SETIME sets the computer clock to an interval corresponding to TNEXT-TLAST appropriately scaled by the speedup factor SPEDUP and the conversion factor for model time units and minutes, TUPM. The maximum interval that can be set is specified by TIMAX.

SIMDAY(ITIME,IDAY,IHOUR,IMIN,ISEC)
converts the simulation time given by 14400*NRESET+ITIME time units to a simulation time of IDAY days, IHOUR hours, IMIN minutes and ISEC seconds, taking into account any hour and minute offsets HROFST and IMOFST

SSWSET Allows the model operator to examine the current status of all 50 sense switches and to set or reset any or all of these switches.

SUBINT This routine will initialize all user specified global variables. Since this function is entirely problem specific the user will have to write the entire routine and add the appropriate VAR declarations in USER.ANE (see 5.5).

UNFRM(A,B,ISTRM)
This function subroutine will generate a uniformly distributed number UNFRM between A and B from random number stream ISTRM.

5.3 Input - Output

Provision has been made to permit at run time to specify the input and output devices for broad categories of inputs and outputs. 20 such categories have been provided through the use of variables DEVICE1 to DEVICE20 to be used as device specifications for input and output.

50 logical variables in the form of array SENSESWITCH are provided to control program logic via the settings of these 'sense switches'. A typical use of sense switches and I/O device specification is the following:

```
IF SENSESWITCH[43] THEN writeln(DEVICE5,I,J)
```

This provides a display of variables I and J only if sense switch 43 is set. Furthermore, the display device is specified by DEVICE5. If this was a CRT terminal, for instance, and the user wants to have a hard copy on a line printer, he only has to change the value of DEVICE5 appropriately.

Some of the input output categories and sense switches have already been assigned as shown in Tables 5.3.1 and 5.3.2. The I/O device variables DEVICE1 to DEVICE20 are initialized in routine ANINIT which reads the values from the data file ANECOM.DAT. To initialize an I/O variable to output to the operator CRT display, specify SYS$OUTPUT in (ANECOM.DAT) as a device for that variable. For input from the keyboard, use SYS$INPUT. All other names refer to disk files. For example if you specify INFO in ANECOM.DAT for a DEVICE variable, the I/O operations will be performed on the disk file INFO.DAT. Most of the device variables will be used for output but at least one (e.g. DEVICE19

SENSE SWITCH #	ACTION IF SWITCH IS SET
1	Display activity, file number and attributes when filing, inserting or removing entries of a GASP file
2	Display overlay module number prior to loading overlay module
3	Display event number and simulation time prior to event execution
4	Echo input data

TABLE 5.3.1 Sense Switch Assignments

DEVICE	CATEGORY DESCRIPTION
DEVICE 1	Error messages due to invalid event numbers, filing array full, or the occasion of resetting the simulation time.
DEVICE 2	Filing array manipulation such as filing, inserting, removing, listing of content.
DEVICE 3	Echo of input data
DEVICE 5	Information display
DEVICE 6	Prompts for operator commands
DEVICE 9	Acceptance of operator commands

TABLE 5.3.2 Input-Output Device Category Assignment

Table 5.3.2) will input operator data. For an input device, you must change the rewrite statement in routine ANINIT for that device to reset. (The rewrite procedure opens a file for output whereas the reset procedure opens a file for input).

5.4 File Assignment

Only the time file has been assigned in this program package. GASP file 1 is the time file. A total of NOQYY GASP files have been provided, allowing the user to assign files 2 to NOQYY as required.

Up to IMZZ attributes have been allowed for each file entry. For the time file the event time has been assigned to attribute 1 and the event number has been assigned to attribute 2. Assignment of attributes 3 to IMZZ are at the user's disposal, as are all attributes for user defined GASP files.

5.5 Global Variables

The ANEVENT package is supported by variables stored in an include file called SYSTEM.ANE. Constant and type declarations are included in this file.

The user has to provide a similar file for any user-defined global variables (e.g. USER.ANE). Both of these files must be included in the compilation of the main program using the %INCLUDE directive.

All subroutines must be declared external in the main program. The declarations are put in a file: ROUTINES.ANE and included in the main program with %INCLUDE. (see listing of main program in List 5.5).

PARAMETERS

IMZZ	Number of attributes
NOQYY	Maximum number of GASP files
NHSTXX	Maximum number of histograms
NCELXX	Maximum number of cells per histogram

5.6 Compilation

In the ANEVENT package, each subroutine is in its own disk file and is compiled separately. To accomplish this, a Pascal environment file is set up in the main program called ANE.PEN. When the main program is compiled, ANE.PEN is created and contains all constant, type, variable and external procedure declarations. Each subroutine file is declared as a MODULE with the actual procedure in this MODULE.

List 5.5 (a) Main Program ANEVENT

```
[ENVIRONMENT('ANE.PEN'), INHERIT('SYS$LIBRARY:STARLET')]
PROGRAM anevent(input, output, anecom, cxc1, dispsq, display, device1, device2,
            device3, device4, device5, device6, device7, device8, device9,
            device10, device11, device12, device13, device14, device15, device16,
            device17, device18, device19, device20, modsa1, modsa2, modsa3, modsa4,
            modsa5, database1, database2, database3, database4, database5,
            database6, database7);

%INCLUDE 'SYSTEM.ANE/NOLIST'

%INCLUDE 'USER.ANE/NOLIST'

%INCLUDE 'ROUTINES.ANE/NOLIST'

BEGIN

   {start initialization and prompt for command}

   mod101;
   cmndrd;

   REPEAT

      {set pointer to first entry of time file}

      ev1 := base[1];
      tnext := ev1^.attributes[1];

      {if next target is equal to the present time, execute next event now}

      IF tnext = tlast THEN iflag := 2
      ELSE modrun;
      IF iflag = 1 THEN

      {operator interrupt occurred}

         BEGIN
            cmndrd;                          {prompt for command}
            $clref(66);                      {clear operator event flag}
            IF mdlmod = interactive THEN tlast := simtim
         END;
      IF iflag = 2 THEN

         {timer interrupt occurred: remove next time file entry and execute}

         BEGIN
            tlast := tnext;
            rmove(1,1);
            event := column[2];
            evnts;
         END

   {loop until time file is empty}

   UNTIL nqzz[1] = 0;
END.
```

List 5.5 (b) Include file: SYSTEM.ANE

```
CONST
   imzz   = 10; {number of file attributes}
   noqyy  = 15; {maximum number of QASP files used}
   nhstxx = 25; {maximum number of histograms}
   ncelxx = 22; {maximum number of cells per histogram}
   icntrx = 14; {number of counters}
   iwrapx = 15; {number of wrappers}
   icashx = 15; {number of cashiers}

TYPE
   eighty = PACKED ARRAY[1..80] OF char;
   six    = PACKED ARRAY[1..6] OF char;
   twelve = ARRAY[1..12] OF integer;
   eleven = PACKED ARRAY[1..11] OF char;
   list = ^object;
   object =
      RECORD
         next       : list;
         attributes : twelve
      END;

VAR
   { anevnt system variables }

   anecom      : text;          {file containing initialisation variables}
   modsa1,modsa2,modsa3,modsa4,modsa5 : text;  {model state files}
   device1,device2,device3,device4,device5,device6,device7,device8,device9,
   device10,device11,device12,device13,device14,device15,device16,device17,
   device18,device19,device20 : text; {output devices}
   ev1         : list;
   esc         : char;          {escape character}
   link        : integer;       {overlay module to be loaded next}
   chan        : [WORD,VOLATILE] -32768..32767;  {QIO channel}
   qioflag     : boolean;       {flag used in MODRUN for asynchronous mode}
   cctime,ccdate: eleven;       {system date and time}
   iseed       : integer;       {first random seed?}
   event       : 1..25;         {number of current event being executed}
   column      : twelve;        {temporary buffer for file attributes}
   tatrib      : twelve;        {temporary buffer}
   senseswitch: ARRAY[1..50] OF boolean; {array for sense switches}
   simtim      : integer;       {simulation time in time units}
   nreset      : integer;       {# of times evt001 has reset time variables}
   tlast       : integer;       {time units elapsed at last event since start}
   tnext       : integer;       {event time of next scheduled event}
   timax       : integer;       {maximum computer clock time}
   speedup     : integer;       {model time speedup factor}
   noq         : 1..noqyy;      {number of QASP files}
   tupm        : real;          {time units per minute of model time}
   hofst,mofst: integer;        {hour and minutes at simulation start}
   nhist       : 1..nhstxx;     {number of histograms desired}
   iflag       : integer;       {general purpose flag used in mod101,anevnt}
   mdlmod      : (interactive,asynchronous); {model mode}
   isd         : ARRAY[1..100] OF integer; {seed for each stream}
   inn         : ARRAY[1..noqyy] OF (high,low); {ranking rules}
   krank       : ARRAY[1..noqyy] OF 1..imzz; {ranking attribute}
   nqzz        : ARRAY[1..noqyy] OF integer; {# of entries in file}
   base        : ARRAY[1..noqyy] OF list;  {base pointers for each file}
```

```
ncels       : ARRAY[1..NHSTXX] OF 0..ncelxx; {number of cells}
ahist,whist: ARRAY[1..NHSTXX] OF real;{lower range and cell width}
jcels       : ARRAY[1..NHSTXX,1..NCELXX] OF integer;
inpbuf      : PACKED ARRAY[1..80] OF char; {input buffer}
error       : boolean;      {error flag for NSCAN and DEVSCN}
iscnst      : 1..80;        {pointer to array element of INPBUF}
iwork       : eleven;
devicecode,devicenumber : integer; {used with INFO command}
```

List 5.5 (c) Include file: USER.ANE

```
{ user variables }

 cxc1          : text;           {data for IRNG routine}
 dispsq        : text;           {sequence chart header}
 display       : text;           {model status display file}
 database1, database2, database3, database4, database5, database6,
 database7     : text;           {file for initializing global user variables}
 cntatr        : ARRAY[1..14,1..11] OF integer; {counter attributes}
 arrdat        : ARRAY[1..2,1..15] OF integer;  {arrival data}
 stfdat        : ARRAY[1..30,1..20] OF integer; {staff data}
 jariv         : integer;        {current arrival data pair number}
 puncdx        : twelve;         {distribution of lateness}
 puncdy        : twelve;         {same}
 itpcx, itpcy: twelve;           {items per customer}
 puncd, itpc   : integer;        {# of readings for puncdx/y and itpcx/y}
 strcls        : integer;        {scheduled store-closing time}
 stfpay        : ARRAY[1..3] OF integer;   {staff pay details}
 novhtm        : integer;        {overhead service time per customer}
 budget        : ARRAY[1..7] OF integer;   {budgeted costs}
 brkdur        : ARRAY[1..4] OF integer;   {break durations}
 wait          : ARRAY[1..5] OF integer;   {customer wait time data}
 iday          : 1..7;           {current day of the week}
 run, rcase    : integer;        {case and run number}
 items         : ARRAY[1..3] OF integer;   {data on items purchased}
 cost          : ARRAY[1..7] OF real;
 seqbuf        : PACKED ARRAY[1..132] OF char;  {buffer for sequence chart}
```

List 5.5 (d) Include file: ROUTINES.ANE

```
{Procedure declarations for main program ANEVNT}

FUNCTION  ran1(stream:integer):real; EXTERNAL;
PROCEDURE rn15(VAR  num:integer); EXTERNAL;
PROCEDURE aninit; EXTERNAL;

PROCEDURE filem(num :  integer); EXTERNAL;
PROCEDURE insert(num:integer;col:integer;ifl:integer); EXTERNAL;
PROCEDURE defile(num:integer;  value:integer;  att:integer); EXTERNAL;
PROCEDURE rmove(element:integer;  num:integer); EXTERNAL;
PROCEDURE dstroy(num:integer); EXTERNAL;
PROCEDURE gfind(value:integer;num:integer;att:integer;
            VAR ele:integer;  VAR buf:twelve); EXTERNAL;
PROCEDURE find2(value1,value2:integer;num:integer;att1,att2:integer;
            VAR ele:integer;VAR buf:twelve); EXTERNAL;
PROCEDURE find3(value1,value2,value3:integer;num:integer;
            att1,att2,att3:integer;
            VAR ele:integer;VAR buf:twelve); EXTERNAL;

PROCEDURE comint; EXTERNAL;
PROCEDURE datan;  EXTERNAL;
PROCEDURE histin; EXTERNAL;
PROCEDURE histog(x1:real;n:integer); EXTERNAL;
PROCEDURE htitle(num:integer); EXTERNAL;
PROCEDURE phist(num:integer); EXTERNAL;
PROCEDURE header(VAR device:text); EXTERNAL;
PROCEDURE bufsca(VAR nbuf:integer); EXTERNAL;
PROCEDURE add; EXTERNAL;
PROCEDURE change; EXTERNAL;
PROCEDURE sswset; EXTERNAL;
PROCEDURE remove; EXTERNAL;
PROCEDURE info; EXTERNAL;
PROCEDURE devscn(n:integer);EXTERNAL;
PROCEDURE mod103; EXTERNAL;
PROCEDURE listfl(num:integer);EXTERNAL;
PROCEDURE quit; EXTERNAL;
PROCEDURE save; EXTERNAL;
PROCEDURE savest(VAR temp:text);EXTERNAL;
PROCEDURE recall(VAR temp:text); EXTERNAL;
PROCEDURE cmderr; EXTERNAL;
PROCEDURE nscan(ibuf:eleven;n:integer;VAR number:integer);EXTERNAL;
PROCEDURE evnts; EXTERNAL;
PROCEDURE mod101; EXTERNAL;
PROCEDURE mod102; EXTERNAL;
PROCEDURE mod106; EXTERNAL;
PROCEDURE cmndrd; EXTERNAL;
PROCEDURE simday(itime :  integer;
             VAR iday,ihour,imin,isec:integer); EXTERNAL;
PROCEDURE irng(ix,kumx:twelve;ncx,istrm:integer;
            VAR iout:integer); EXTERNAL;
PROCEDURE bufprt(VAR temp :  text); EXTERNAL;
PROCEDURE miscom; EXTERNAL;
PROCEDURE setime; EXTERNAL;
PROCEDURE modrun; EXTERNAL;
PROCEDURE evt001; EXTERNAL;

     {user routines}
```

```
PROCEDURE sqhead(VAR temp:text); EXTERNAL;
PROCEDURE subint(VAR datbas : text); EXTERNAL;
FUNCTION  itu(ftime:integer):integer; EXTERNAL;
PROCEDURE crt11;  EXTERNAL;
PROCEDURE crt12(counter:integer); EXTERNAL;
PROCEDURE crt13(staff:integer); EXTERNAL;
PROCEDURE screen(mode,row,col:integer); EXTERNAL;
PROCEDURE evt002; EXTERNAL;
PROCEDURE evt003; EXTERNAL;
PROCEDURE evt004; EXTERNAL;
PROCEDURE evt005; EXTERNAL;
PROCEDURE evt006; EXTERNAL;
PROCEDURE evt007; EXTERNAL;
PROCEDURE evt008; EXTERNAL;
PROCEDURE evt009; EXTERNAL;
PROCEDURE evt010; EXTERNAL;
PROCEDURE evt012; EXTERNAL;
PROCEDURE evt013; EXTERNAL;
PROCEDURE evt014; EXTERNAL;
PROCEDURE evt015; EXTERNAL;
PROCEDURE evt016; EXTERNAL;
PROCEDURE evt017; EXTERNAL;
PROCEDURE evt018; EXTERNAL;
PROCEDURE evt019; EXTERNAL;
PROCEDURE evt020; EXTERNAL;
PROCEDURE evt021; EXTERNAL;
PROCEDURE evt022; EXTERNAL;
PROCEDURE evt023; EXTERNAL;
PROCEDURE evt024; EXTERNAL;
PROCEDURE evt025; EXTERNAL;
PROCEDURE doper;  EXTERNAL;
PROCEDURE cstat1(counter:integer;normal:integer;manning:integer);EXTERNAL;
PROCEDURE sstat1(staff:integer;ibstat:integer;loc:integer;avail:integer);
          EXTERNAL;
```

The module inherits ANE.PEN as an environment and can now access the global variables and procedures defined in the main program. For linking, each procedure is preceded by the compiler directive [GLOBAL].

VARIABLES

CHAN — The channel number assigned by the VAX 11/780 systems services routine QIO to the terminal from which the model operator issues commands.

IMZZ — Maximum number of attributes used for a file entry.

NOQ — Number of GASP files used.

ISEED — Not used

COLUMN — Integer attribute transfer array used to enter contents of COLUMN into a file entry or to extract the contents of a file entry into COLUMN.

INN(I) — Specifies the ranking rule for file I = low means lowest value first = high means highest value first.

KRANK(I) — Specifies the attribute number of an entry in file I which determines ranking in the file.

NQZZ(I) — This provides the number of entries currently in file I.

TATRIB(I) — Integer array of same length as COLUMN to be used as a scratch storage. Typically used as buffer array in routines GFIND, FIND2 and FIND3 to hold a copy of the entry found.

TLAST — An integer variable containing the number of time units elapsed at the last event time since the start of simulation.

TNEXT — An integer variable containing the event time of the next scheduled event in time units.

TIMAX — An integer variable containing the maximum time in time units that will be set on the computer clock.

SIMTIM — Integer variable to track the current simulation time in time units since the start of the simulation. Simulation time equals SIMTIM + 14400 * NRESET.

SPEEDUP — An integer variable representing the model time speedup factor.

TUPM — Time units per minute of model time. (ie. TUPM=10. means that 1 minute of model time corresponds to 10 time units as counted for example by SIMTIM)

HOFST Integer number specifying the hour at which the simulation is starting.

MOFST Integer number specifying the minute at which the simulation is starting. (ie at the start of simulation SIMTIM=0, NRESET=0 corresponds to a model time of HOFST hours and MOFTST minutes).

NRESET The number of times all time variables in the model have been reset by 14400 time units in routine EVT001. Note that on a 32 bit computer it is not necessary to reset time, hence NRESET will always equal zero.

EVENT The number of the event to be executed next.

BASE[1] An array of base pointers for the GASP files.

DEVICECODE,DEVICENUMBER
(used internally by routine INFO)

QIOFLAG Flag used for the asynchronous mode.

IFLAG General purpose variable, used for instance to determine type of initialization needed in the initialization module or to determine the type of interrupt received.

LINK Variable indicating the overlay module to be loaded next. On return from an overlay module LINK specifies the next module to be loaded or, if LINK=0, requests a return to the main module.

MDLMOD A variable indicating the model mode. MDLMOD=interactive specifies the synchronous mode while MDLMOD=asynchronous specifies the asynchronous mode.

CCDATE Contains the date at the computer site at the start of a simulation run.

CCTIME Contains the time at the computer site at the start of a simulation run.

SENSESWITCH[I]
logical array element storing the status of sense switch I.

INPBUF(I) An 80 element input buffer for reading operator commands in character format.

IWORK(I) A 10 element array mainly used for temporary storage of a single command field extracted from INPBUF.

ISCNST A pointer to an array element of INPBUF where field extraction is to begin.

NHIST Specifies the number of histograms desired. Note that NHIST can not exceed NHSTXX.

NCELS(I) Specifies the number of cells in histogram I. This includes the boundary cells for counting values that are outside the histogram range.

JCELS(I,J) Stores the count for cell J in histogram I.

AHIST(I) A real value which specifies the lower range of histogram I.

WHIST(I) A real value which specifies the cell width of histogram I.

ISD(I) The current random number seed for random number stream I.

DEVICE1...DEVICE20
Input/output files for CRT terminals, printers, disk files etc. initialized in routine ANINIT.

6.0 CREATION OF THE ANEVENT LIBRARY

The ANEVENT library ANELIB contains all programs listed in section 5.2 with the exception of the main program ANEVENT. This library has to be recreated every time a change is made to the main program.

On the VAX 11/780 this is accomplished by issuing the statement:

```
@ANEBLD
```

This will accomplish the following:

- execute SUBLIST.COM which compiles the main program and all the subroutines needed.
- Assemble assembly program RN15.MAR to produce RN15.OBJ.
- Create a library ANELIB and insert all object modules into it.

The content of files ANEBLD.COM and SUBLST.COM are listed in List 6.1.

7.0 DATA FOR GLOBAL VARIABLES IN SYSTEM.ANE

Initial values for some of the variables in the file SYSTEM.ANE are read in during initialization from a data file

List 6.1 (a) Command Procedure to Create ANEVENT Library

```
!!!
```

ANECOM.DAT. List 7.0 shows the content of this data file for the declarations shown in fig. 5.5. Note that the text portions represent comments which are not ready by the program. Table 7.0 shows which routine is used for reading the various data items.

8.0 TRANSFER OF ANEVENT PACKAGE TO ANOTHER COMPUTER

The ANEVENT package used for this documentation was run on a VAX 11/780 digital computer. Use was made of VAX systems services routines and a machine specific random number generator which may not be available on other computers. The following is a list of potential difficulties where changes may be required in order to be able to run the package on other computers.

8.1 Compilation

Environment files are used to allow separate compilations of the routines as described in 5.6. The %INCLUDE directive was used in the main program. It tells the compiler that the included file is part of the program. Without this directive, the contents of the file SYSTEM.ANE USER.ANE and ROUTINES.ANE must be inserted directly into the program. These VAX-11 Pascal features will have to be adapted.

8.2 Random Number Generation

Subroutine RAN1 calls a uniform random number generating subroutine RN15 which is coded in assembly language. RN15 produces odd, uniformly distributed numbers between 1 and 32767 using an integer seed argument [See Ref. 2].

8.3 I/O Device Initialization

The DEVICEn text variables are assigned to a specific device by reading in the name of the device in routine ANINIT. The OPEN statement is used to assign the name read in to the text variable DEVICEn. (see 5.5).

8.4 Reading Time of Day and Date

In ANINIT line 70, calls to subroutines DATE and TIME are made to obtain the computer site date and time which are stored in CCDATE and CCTIME respectively.

8.5 Operator Interrupts and Time Keeping

VAX-11/780 system services routines have been used to handle operator interrupts and time keeping. Every module that uses system services inherits the environment file: SYS$LIBRARY: STARLET.PEN along with ANE.PEN from the main program. The routines that will have to be

List 7.0 Data File: ANECOM.DAT

```
device1 to device 20:
SYS$OUTPUT
QASP
ECHO
SYS$OUTPUT
INFO
SYS$OUTPUT
SYS$OUTPUT
SYS$OUTPUT
SYS$OUTPUT
TEST
ASEQUENCE
SEQUENCE
SYS$OUTPUT
REPORT
SYS$OUTPUT
SYS$OUTPUT
SYS$OUTPUT
SYS$OUTPUT
SYS$INPUT
TIMER
senseswitch:
FALSE FALSE  FALSE FALSE  FALSE FALSE FALSE  FALSE  FALSE FALSE
FALSE FALSE  FALSE FALSE  FALSE FALSE FALSE  FALSE  FALSE FALSE
FALSE FALSE  FALSE FALSE  FALSE FALSE FALSE  FALSE  FALSE FALSE
FALSE FALSE  FALSE FALSE  FALSE FALSE FALSE  FALSE  FALSE FALSE
FALSE FALSE  FALSE FALSE  FALSE FALSE FALSE  FALSE  FALSE FALSE
0 0 4000 0 1000 15 9 0  tlast, tnext, timax, simtim, speedup, tupm, hofst,
asynchronous            mdlmod                                      imofst
2                       nhist
ncels:
20 20
ahist:
0 0
whist:
3 1
isd:
5 1325 3 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
0    0 0 0 0 0 0 0 0 0
15 1                    noq, iseed
1 7 7 7 7 7 7 7 7 7 7 7 7 7 7   krank
low low low low low low low low low low low low low low low low low     inn
-1                      start of entry initialization
0 0 0 0 0 0 0 0 0 0
1                       file event 2 in time file
0 2 0 0 0 0 0 0 0 0
0                       end of entry initialization
0 0 0 0 0 0 0 0 0 0
```

DATA ITEMS	ROUTINE WHICH READS THE DATA
I/O device variables	ANINIT
SENSESWITCH	ANINIT
TLAST,TNEXT,TIMAX,SIMTIM SPEEDUP,TUPM,HOFST,MOFST	COMINT
MDLMOD	COMINT
NHIST	HISTIN
NCELS	HISTIN
AHIST	HISTIN
WHIST	HISTIN
ISD	COMINT
NOQ,ISEED	DATAN
KRANK	DATAN
INN	DATAN
JQ, COLUMN*	DATAN

* A JQ=-1 indicates the start of GASP file entry intitialization, a JQ=0 terminates GASP file entry initialization. All intermediate entries specify the file number JQ and the attributes of the entries to be initialized.

TABLE 7.0 GLOBAL VARIABLE INITIALIZATION

modified are: ANINIT,ANEVENT, MODRUN,QUIT,SETIME and CMNDRD. [see the listings of these for additional documentation].

8.6 Non-Standard Functions and Procedures

All non-standard features in the ANEVENT package with the exception of the OPEN procedure are used in connection with operator interrupts and time keeping.

i) SUBSTR(s,b,l) s=string b=integer l=integer

forms a new string starting at position b of string s and extending to length l.

- used in MODRUN lines 19, 20, 35, 36 to extract the "seconds" digits from the time string.

- note: the variable assigned to this function must be of type: VARYING which is a string variable of varying length.

ii) UAND(U1,U2) UOR(U1,U2)

perform binary logical AND or OR on the corresponding bits of U1 and U2.

UAND is used in MODRUN line 30 to determine which event flag was set. UOR is used in SETIME, line 32 for the QIO call.

iii) INT(X)

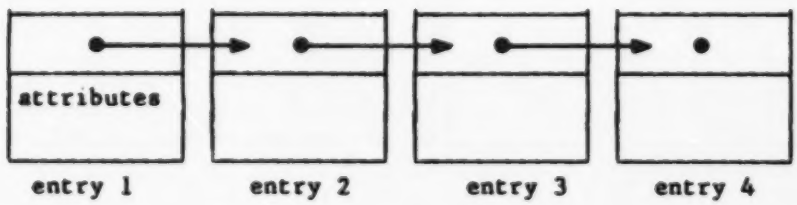
converts value of x to an integer

used in SETIME, line 32 for the QIO call.

9.0 DIFFERENCES BETWEEN THE FORTRAN AND PASCAL VERSIONS

i) GASP file manipulation:

GASP files in the ANEVENT package are organized as linked lists which each element of a list forming an entry with attributes. In Pascal, linked lists are handled with pointer variables. Each entry has an array for attributes and a pointer to the next entry.



In this way, insertion and deletion in the list is accomplished by simply changing the pointers to point to or bypass an entry. Each linked list has a base pointer which always points to the first entry in that list. To access a specific entry in a list, you set up a pointer variable that points to the first entry and advance step by step through the list. The position of an entry in a GASP file corresponds to its position in the linked list. For example, to remove the first entry in the time file, you point a pointer variable to the first entry in the corresponding linked list and access its attributes. The entry is then removed by pointing the base pointer of the list to the next entry which now becomes the first entry. With pointer variables, the code for file manipulation is simplified. This allows the file manipulation routines such as FILEM and RMOVE to make pointer changes directly, eliminating the need for a routine SET mentioned in reference 1. [For a good introduction to linked lists in Pascal, see Chapter Eight of Programming in Pascal, Peter Grogono, Addison-Wesley 1980]. See the code for more detailed documentation].

ii) Global Variables

In FORTRAN, common areas must be set up for global variables and are included in every routine where they are required. In Pascal, all variables declared in the main program become global to all routines.

These two differences mentioned here are perhaps the most important ones to note.

10 REFERENCES

1. Pritsker A.A.B. 'Simulation with GASP II, A Fortran Based Simulation Language'. Prentice-Hall Inc. 1969.

2. Phan C.S. 'Generation of Random Samples Using Non-Parametric Methods'. NRC Control Systems and Human Engineering Report LTR-CS-199, March 1979.

3. Crate G.F. Ltd. 'ANEVENT: An Interactive Computer Modelling Package Based on Discrete Event Simulation - A User's Guide'. NRC Analysis Laboratory Report LTR-AN-53, Oct. 1982.

4. Pritsker A.A.B. 'The GASP IV Simulation Simulation Language'. John Wiley and Sons Inc. 1974.

REPORT DOCUMENTATION PAGE/PAGE DE DOCUMENTATION DE RAPPORT

REPORT/RAPPORT TR-SYS-001 1a	REPORT/RAPPORT NRC No. 22939 1b
REPORT SECURITY CLASSIFICATION CLASSIFICATION DE SÉCURITÉ DE RAPPORT UNCLASSIFIED 2	DISTRIBUTION/DIFFUSION UNLIMITED 3

TITLE/SUBTITLE/TITRE/SOUS-TITRE (CLASSIFICATION) (Unclassified)
ANEVENT - An Interactive Computer-Modelling Package Based on Discrete Event Simulation: Pascal Version
4

AUTHOR(S)/AUTEUR(S)
Carl Angers, P.W.U. Graefe
5

SERIES/SÉRIE
Technical Report
6

CORPORATE AUTHOR/PERFORMING AGENCY/AUTEUR D'ENTREPRISE/AGENCE D'EXÉCUTION
7

SPONSORING AGENCY/AGENCE DE SUBVENTION
8

DATE	FILE/DOSSIER	LAB. ORDER COMMANDE DU LAB.	PAGES	DIAGS	REFS
1983/12			(iv) 45	7	4
9	10	11	12a	12b	12c

NOTES
13

DESCRIPTORS(KEY WORDS)/MOTS-CLÉS
14

SUMMARY/SOMMAIRE
Describe the ANEVENT Package - see title
15

ADDRESS/ADRESSE
P.A. Hamill, Head
Systems Laboratory
Division of Mechanical Engineering
National Research Council Canada
Montreal Road, Ottawa, Ontario K1A 0R6 (613-993-9208)
16